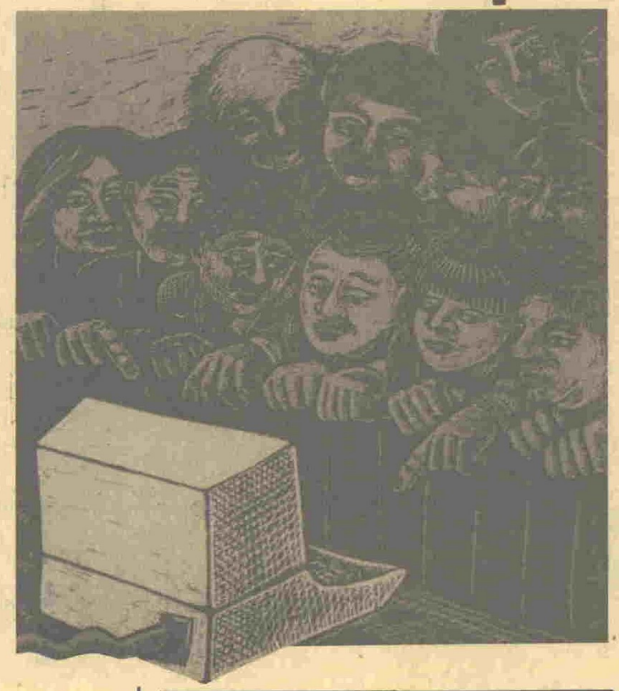
SOFTMAIL

โปรแกรมเพื่อคุณ



บุญเลิศ เอี่ยมทัศนา

จากวงสนทนามีผู้ตั้งปัญหาถามผมว่าจะเขียนโปรแกรมอ่านไฟล์ของเรคอร์ดต่างแบบกันให้ได้ทุกไฟล์ได้ไหม ผมตอบไปว่าทำได้แค่เขียนโดยตรงเฉพาะไฟล์สะดวกกว่า การสนทนาดำเนินการต่อไปจนกลายเป็นว่า ต้องการโปรแกรมสำหรับให้ผู้ที่คุ้นเคยกับคอมพิวเตอร์เพียงเล่นเกมได้ สามารถนำไปใช้ในงานบัญชีเล็ก ๆ น้อย ๆ ได้ ผมแย้งต่อไปว่าให้ใช้ดีเบส ซึ่งหากความต้องการเพียงเท่านี้ใช้เวลาศึกษาไม่กี่ชั่วโมงก็น่าจะใช้งานได้แล้ว คำตอบคือไม่ศึกษา

ผมมานึกดูจึงเห็นว่ายุคสมัยของคอมพิวเตอร์ได้เปลี่ยนไปทีละน้อย ก่อนนี้เขาเขียนโปรแกรมสำหรับผู้ใช้ที่อยู่ในแวดวงของคอมพิวเตอร์ แต่ปัจจุบันและอนาคตเราต้องเขียนโปรแกรมสำหรับผู้ใช้ที่เป็นบุคคลทั่วไป การศึกษาในเรื่องของ Human Interface จึงมีมากขึ้น ซึ่งสำหรับผมจะมองในแง่ของ **"ใช้งานและเป็นกันเอง"** เท่านั้น

จึงนำโปรแกรมในวงสนทนามากล่าวถึง เพราะโปรแกรมนี้สามารถเขียนให้ใช้ง่ายและเป็นกันเองได้ แต่ผมจะเขียนในส่วนที่เป็นหลักการเท่านั้น การใช้ง่ายและเป็นกันเองเป็นเรื่องที่คุณจะต้องดำเนินการเอง (โดยที่ผมจะแสดงความคิดเห็นของผมให้ทราบ) จึงตั้งชื่อเรื่องนี้ **"โปรแกรมเพื่อคุณ"** และตั้งชื่อโปรแกรมว่า For You ซึ่งผมพูดแทนคุณครับ

สำหรับหลักการในเรื่องนี้ก็คือการเขียนโปรแกรมให้ผู้ใช้กำหนดฟิลด์ของเรคอร์ดได้ตามชอบ โดยไม่ต้องรู้จักภาษาคอมพิวเตอร์ใดเลย เช่น ผู้ใช้ต้องการทำบัญชีรายจ่ายก็จะกำหนดฟิลด์เป็นวันที่ รายการ จำนวนเงิน และหมายเหตุ ซึ่งก็คือการใช้งานเบื้องต้นของดีเบสนั่นเอง และการเขียนโปรแกรมนี้ก็เป็นเบื้องต้นของการเขียนโปรแกรมประเภทฐานข้อมูลเช่นกัน

ใช้ง่ายและเป็นกันเอง

ความใช้ง่ายขึ้นอยู่กับความต้องการของมนุษย์ ผมไม่ทราบว่าถ้าเราพัฒนาต่อไปจนสามารถสั่งการคอมพิวเตอร์ได้ด้วยวาจา เช่น ฟอร์แมตดิสก์ ขอดูไดเรคทอรี พิมพ์ไฟล์ ABCD มาดูกันหน่อยซิ ระหว่างคอยเครื่องพิมพ์ ให้คิดเลข...ให้ด้วย จะเป็นที่พอใจว่าใช้ง่ายหรือไม่

ในปัจจุบันที่ใช้กันโดยทั่วไปได้แก่ระบบกดปุ่มและเลื่อนเคอร์เซอร์

- กดปุ่ม คือที่ให้กดคีย์หน้าที่พิเศษต่าง ๆ ซึ่งถ้ามากเกินไปผู้ใช้จำไม่ไหวก็ต้องแสดงหน้าที่ของคีย์ให้ดูได้บ้างไม่ได้บ้าง ในที่สุดก็กลายเป็นใช้ไม่ง่ายไป ในโปรแกรมนี้ผมจึงจะใช้คีย์ให้น้อยที่สุด
- เลื่อนเคอร์เซอร์ ซึ่งอาจทำได้ด้วยคีย์ลูกศรหรือใช้เมาส์ เพื่อเลื่อน

เคอร์เซอร์ไปยังรายการที่ต้องการแล้วกดคีย์อื่น เช่น Enter ซึ่งถ้ามากเกินไปอย่างเทอร์โบปาสคาลผู้ใช้ก็ลานตากลายเป็นใช้ไม่ง่ายไปอีกเช่นกัน

อีกเรื่องหนึ่งของการใช้ง่ายคือ มีคำอธิบายให้ทราบว่าจะทำอะไรได้อย่างไร ซึ่งถ้าให้แสดงเมื่อเริ่มต้นรันโปรแกรม ผู้ใช้จะรู้สึกตื่นเต้นพอใจ แต่เมื่อรันโปรแกรมนี้ไปนาน ๆ ก็จะเบื่อที่ต้องเห็นคำอธิบายทุกครั้ง ก็ต้องเลี่ยงไปเป็นให้ดูเมื่อต้องการเท่านั้น

ในเรื่องของความเป็นกันเองผมเห็นว่าส่วนหนึ่งคือ ถ้อยคำที่ใช้โต้ตอบกับผู้ใช้ให้เป็นกันเอง ซึ่งสำหรับเราคงหนีไม่พ้นภาษาไทย ในโปรแกรมของผมใช้ภาษาอังกฤษ ขอให้เปลี่ยนเป็นภาษาไทยด้วยครับ (ซึ่งขึ้นอยู่กับระบบภาษาไทยของแต่ละเครื่องซึ่งไม่เหมือนกันไม่เป็นหนึ่งเดียว)

อีกส่วนหนึ่งคือ คอยช่วยเหลือผู้ใช้หากดำเนินการใดผิดก็ให้สามารถแก้ตัวได้ ไม่ใช่เผลอป้อนอักษรแทนตัวเลขเท่านั้น ถือว่าเป็นความผิดมหันต์ออกจากโปรแกรมไปเลย ในโปรแกรมของผมไม่ได้ดำเนินการไว้ แต่จะชี้ให้เห็น

เริ่มที่ปาสคาล

เนื่องจากโครงงานนี้เริ่มจากความต้องการที่จะอ่านไฟล์ข้อมูลในภาษาปาสคาล และผมก็เริ่มจากการศึกษาการเก็บข้อมูลในภาษาปาสคาล (มาก่อน) เช่นกัน จึงเห็นว่าสมควรจะนำมากล่าวถึงก่อนเพื่อเป็นพื้นฐานในการทำความเข้าใจ กล่าวคือ

หากเรากำหนดตัวแปรเรคอร์ดและไฟล์ของเรคอร์ดตามรูปที่ 1 ปาสคาลจะจัดเนื้อที่ในหน่วยความจำ 49 ไบต์ให้แก่ตัวแปร perrec เพื่อเก็บสตริง 41 ไบต์ เลขจำนวนเต็มบวก 2 ไบต์ และเลขจำนวนจริง 6 ไบต์ เรียงติดต่อกันไป

เมื่อนำไปเก็บในไฟล์ pfi ก็จะใช้เนื้อที่ในดิสก์ 41 ไบต์สำหรับ 1 เรคอร์ดเช่นกัน และหากมีเรคอร์ดต่อ ๆ ไปก็จะเก็บเรียงติดต่อกันไปเรคอร์ดละ 41 ไบต์

และหากเราโหลดไฟล์นี้ในฐานะ ARRAY [0...X] OF Byte จะได้ข้อมูลเช่นเดียวกับที่อยู่ในดิสก์แต่อยู่ในหน่วยความจำแทนโดยที่ อะเรย์ช่องที่ 0 ถึง 40 คือสตริง ช่องที่ 41 กับ 42 คือเลขจำนวนเต็มบวก และช่องที่ 43 ถึง 48 คือเลขจำนวนจริง แล้วเริ่มต้นเรคอร์ดที่ 2 ที่ช่อง 49 ใหม่

ดังนั้น หากเรารู้ว่าในไฟล์ของเรคอร์ดนั้นมีกี่ฟิลด์ และในแต่ละฟิลด์เก็บข้อมูลแบบใด (หากเป็นสตริงจะต้องรู้ความยาวของสตริงด้วย) ก็สามารถอ่านไฟล์ของเรคอร์ดแบบใดก็ได้มาดำเนินการใดก็ได้(ความจริงดำเนินการได้กับไฟล์ทุกแบบ เช่น ไฟล์ของเลขจำนวนเต็มก็คือไฟล์ของเรคอร์ดที่มีฟิลด์เลขจำนวนเต็มเพียงฟิลด์เดียว) สำหรับค่า X คือขนาดของไฟล์เป็นไบต์ลบด้วย 1 (เพราะอะเรย์เริ่มจาก 0)

ในการดำเนินการหากผมให้ idx, idxx และ recnum เป็นตัวแปรแบบ Word เพื่อใช้ idx ชี้ที่ต้นเรคอร์ด idxx ชี้ที่ต้นฟิลด์ และ recnum หมายถึงเรคอร์ดที่ n โดยให้เริ่มจากศูนย์ จากตัวอย่างที่ยกมาผมจะหาค่า idx ได้จาก idx: =recnum * 49 ส่วนค่าของ idxx สำหรับฟิลด์ name จะเป็น idx + 0 ของฟิลด์ age เป็น idx + 41 และของฟิลด์ high เป็น idx + 41 + 2 ผมจะสามารถอ่านค่าในเรคอร์ดใดฟิลด์ใดก็ได้ตามจำนวนไบต์ซึ่งขึ้นกับแบบของข้อมูลมาดำเนินการได้

นั่นคือ ผมต้องเก็บเรื่องราวของแต่ละฟิลด์ไว้ในตัวแปร ซึ่งต้องเป็นแบบเรคอร์ด (ตรงนี้ต้องระวังครับ เพราะเป็นเรคอร์ดซ้ำเรคอร์ด) ดังที่กำหนดเป็นแบบข้อมูล FieldT ในรูป A1 ซึ่งมีเรื่องอื่นนอกจากที่กล่าวมาแล้วอีกคือ

รูปที่ 1 เรคอร์ดในภาษาปาสคาล

```
TYPE    PerRecT  = RECORD
          name   : String[40];
          age    : Word;
          weight : Real;
                   END;

VAR     perrec : PerRecT;
        pfi    : FILE OF PerRecT;
```

A1

```
PROGRAM  ForYou;
uses     crt,dos;
CONST    MAXBUF = 65521;
         Esc = #27;  PgUp = #73;  PgDn = #81;
         NTI = 0;  NFD = 1;  RSI = 2;  NREC = 4;
         SS = 6;  DS = 8;  TIT = 10;
TYPE     FieldT   = RECORD
           head   : String[29];
           dT     : Char;
           deci   : Byte;
           spc    : Byte;
           leng   : Byte;
           offs   : Word;
                    END;
         Str79 = String[79];  Str7 = String[7];
         StrArray = ARRAY[1..2] OF FieldT;
         DataT = ARRAY[0..1000] OF Byte;
         TitArray = ARRAY[1..2] OF Str79;
         TotalT = ARRAY[1..2] OF Real;
         TitleT = ^TitArray;
         StructT = ^StrArray;
         CharSet  = SET OF Char;
VAR      data : ^DataT;  title : TitleT;
         struct : StructT;
         sum    : ^TotalT;
         FiName : Str79;  outfi : Text;
         numti,numfd : Byte;
         recsi,numrec,strstt,datstt : Word;
         line,verline : Str79;
         save,stop : Boolean;
```

● head ไม่เกี่ยวกับการอ่านไฟล์ แต่เกี่ยวกับการแสดงผล เพื่อให้ทราบว่าเป็นข้อมูลของอะไร ซึ่งอาจจะตรงหรือไม่ตรงกับชื่อฟิลด์จริงก็ได้ เช่น ฟิลด์ name อาจให้ค่าแก่ head เป็น Name หรือ Studentname ก็ได้ แต่ต่อไปผมจะเรียกสตริงที่จะให้แก่ head นี้ว่าชื่อฟิลด์ ในฐานะที่เป็นฟิลด์ของผู้ใช้

● dt เพื่อบ่งบอกว่าเป็นข้อมูลแบบใดดังกล่าวมา

● deci เกี่ยวกับการแสดงผล คือถ้าเป็นข้อมูลแบบเลขจำนวนจริงต้องการทศนิยมกี่ตำแหน่ง

● spc เป็นเรื่องของโปรแกรม ForYou ดังจะกล่าวต่อไป

● leng คือความยาวของข้อมูลดังกล่าวมา ซึ่งถ้าเป็นสตริงจะต้องบอกขนาดของสตริง ถ้าเป็นข้อมูลมาตรฐานจะให้โปรแกรมดำเนินการให้

● offs คือค่าที่จะบวกกับ idx เป็น idxx ของฟิลด์นี้ ให้คำนวณมาเก็บไว้เลยแทนที่จะต้องคำนวณทุกครั้ง

หากเรากำหนดตัวแปรตรงตามแบบข้อมูลนี้ก็จะได้เรื่องราวของฟิลด์เพียงฟิลด์เดียว จึงได้กำหนดแบบข้อมูล StrArray เป็นอะเรย์ขนาด 2 ช่องของ FieldT ซึ่งถ้ากำหนดตัวแปรตรงตามนี้ก็จะสำหรับ 2 ฟิลด์เท่านั้น จึงได้กำหนดแบบข้อมูล StructT ให้เป็นพอยน์เตอร์ของ StrArray อีกทอดหนึ่ง แล้วนำไปกำหนดตัวแปรพอยน์เตอร์ Struct

การดำเนินการเช่นนี้ความหมายของ [1...2] รวมทั้งที่กำหนดให้แก่แบบข้อมูลแบบอื่นด้วย จึงเพียงแต่บอกคอมพายเลอร์ว่า Struct เป็นพอยน์เตอร์ไปยังข้อมูลแบบอะเรย์เท่านั้น ส่วนจำนวนช่องจริงขึ้นอยู่กับที่เราจะจองหน่วยความจำด้วย GetMem ต่อไป ทั้งนี้เพราะต้องการเขียนโปรแกรมให้อ่านไฟล์ของเรคอร์ดได้ทั่วไป ไม่จำกัดจำนวนฟิลด์

สำหรับตัวแปรอื่น ๆ ผมขอเว้นไว้ก่อน ที่สมควรจะกล่าวถึงในตอนนี้คือ data เป็นตัวแปรที่เราจะโหลดข้อมูลจากไฟล์มาลง ที่กำหนดจำนวนช่องไว้มากมายเช่นนั้นก็เพื่อให้คุณใช้ดีบักตรวจสอบการเก็บข้อมูลว่าเป็นไปตามที่ผมกล่าวมาหรือไม่ หากไม่สนใจจะให้เป็นอะเรย์ [0...1] ได้เช่นกัน

ขั้นตอนการทำงานต่อไปคือให้ผู้ใช้ (ซึ่งอาจเป็นตัวคุณเอง) ป้อน ชื่อไฟล์ จำนวนฟิลด์ ชื่อฟิลด์ แบบข้อมูล ฯลฯ อ่านมาดำเนินการ ซึ่งทำได้หลายแบบสำหรับในโปรแกรม ForYou ผมจะให้ดำเนินการดังจะกล่าวต่อไปนี้ คือ

ทำให้ง่ายเข้า

ดังได้กล่าวแล้วว่าโปรแกรม For You เขียนขึ้นเพื่อให้ใช้ง่ายสำหรับงานง่าย ๆ จะได้แก่บัญชีหรือตารางที่สามารถแสดงข้อมูลของ 1 เรคอร์ดได้ใน 1 บรรทัด ดังเช่นในรูปที่ 2

จะเห็นว่าถ้าให้ผู้ใช้ป้อน จำนวนฟิลด์ ชื่อฟิลด์ ฯลฯ ก็ไม่ต้องคิดเขียนโปรแกรมนี้หรอกครับใช้ดีเบสดีกว่า ผมจึงออกแบบให้ผู้ใช้ป้อนดังรูปที่ 3 คือ

ร้านสมส่วน

รายการลูกค้า

ที่	ชื่อ	อายุ	น้ำหนัก	โทรศัพท์	หมายเหตุ

รูปที่ 2 แสดงตัวอย่างบัญชีที่ผู้ใช้ต้องการ

```
Enter data type

Enter file name CLIENT.32
How many line of title 2
Enter title
1 SOMSUAN Co.                                                R  60
2 CLIENT MEMO                                                C
Enter design table
==============================================================================
No |Name                        |age  |weight  |tele    |remark              |
==============================================================================

Data type:- S = String, C = Char, I : Integer, R = Real, N = Number

No                                    N
Name                                  S
age                                   I
weight                                R 2
tele                                  S
remark
```

รูปที่ 3 แสดงการกำหนดเรคอร์ดของผู้ใช้คือ ตรงบรรทัดที่มีคำว่า No. Name และอื่น ๆ กับการกำหนดแบบข้อมูลที่ตามมา

เฉพาะตรงตาราง เว้นที่เป็นข้อความว่า ร้านสมส่วนกับรายการลูกค้า ซึ่งผมเรียกว่า ไตเติล ไว้ก่อน

ความหมายคือ ให้ผู้ใช้ป้อนชื่อฟิลด์ชิดซ้าย เว้นช่องว่างไว้ตามแต่จะเห็นควร ใช้เครื่องหมายขีดตั้งปิดเป็นขอบเขตของฟิลด์หนึ่ง ๆ แล้วดำเนินการกับฟิลด์อื่น ๆ ต่อไป (แต่ต้องไม่เกินบรรทัด) และเพราะเป็นการป้อนสตริง ผู้ใช้จึงอาจลบเปลี่ยนแปลงใด ๆ ก็ได้ จนกว่าจะกดคีย์ Enter

นำมาทำเป็นโครงสร้างข้อมูล

ผมควรจะกล่าวว่านำมาทำเป็นเรคอร์ด เพราะเรื่องราวที่กล่าวมาและที่กล่าวในหัวข้อนี้เทียบได้กับการกำหนดตัวแปรแบบเรคอร์ดเท่านั้น แต่เพื่อหนีคำว่า เรคอร์ด จึงใช้คำว่าโครงสร้างข้อมูลแทน

จากสตริงที่ผู้ใช้ป้อนนี้ผมอ่านมาเก็บไว้ในตัวแปร rstr ในโพรซีเดอร์ GetStruct รูป B5 (คือตรงหลังคำสั่ง Writeln (verline) ซึ่งเป็นการให้ตีเส้นบรรทัดคู่) เมื่อนับจำนวนขีดตั้งจะได้จำนวนฟิลด์ในตัวแปร numfd ตามตัวอย่างในรูปที่ 3 จะได้ numfd เท่ากับ 6

นำไปกำหนดเนื้อที่ในหน่วยความจำให้แก่ตัวแปร Struct ในโพรซีเดอร์ GetHead รูป B3 ซึ่งโพรซีเดอร์นี้จะแยกข้อความในแต่ละช่องดังตัวอย่างในรูปที่ 3 เป็นชื่อฟิลด์ให้แก่ head ของฟิลด์ต่าง ๆ และจะได้ค่า spc ด้วย ค่า spc หมายถึง ความกว้างของช่องในตารางตามที่ผู้ใช้กำหนดด้วยขีดตั้ง

เรื่องราวของฟิลด์ที่เราต้องการทราบจากผู้ใช้ต่อไปคือ ในแต่ละฟิลด์หรือคือแต่ละช่องในความคิดของผู้ใช้โปรแกรม ForYou นี้จะให้เก็บข้อมูลจะให้เก็บข้อมูลแบบใด ผมให้โพรซีเดอร์ GetType รูป B4 ดำเนินการทีละฟิลด์

B5

```
PROCEDURE  GetStruct;
VAR     rstr : Str79;  yy,cx,fd : Byte;
BEGIN

  ClearPointer;
  GetTitle(title,numti);  strstt := TIT + 80 * numti;

  WriteLn('Enter design table ');  yy := WhereY + 1;
  verline[0] := Chr(79);
  WriteLn(verline);  WriteLn;  WriteLn(verline);
  GotoXY(1,yy);   ReadLn(rstr);   GotoXY(1,yy+3);
  verline[0] := rstr[0];  numfd := 0;
  for  cx := 2 to Length(rstr)  do
    if  rstr[cx] = '|'  then  inc(numfd);
  datstt := strstt + SizeOf(FieldT) * numfd;
  GetHead(rstr,numfd,struct);
  GetType(numfd,struct);

  recsi := 0;
  for  fd := 1 to numfd  do
  with  struct^[fd]  do
    begin  offs := recsi;  recsi := recsi + leng  end;
END;
```

B3

```
PROCEDURE  GetHead(rstr : Str79;nfd : Byte;
                   var struct : StructT);
VAR     fd,cx,cb,ca,len : Byte;
BEGIN
  GetMem(struct,SizeOf(FieldT)*nfd);  cx := 1;
  FOR  fd := 1 TO nfd  DO
  WITH  struct^[fd]  DO
    begin
    ca := cx;
    while  rstr[cx] <> '|'  do  inc(cx);
    spc := cx - ca;  cb := cx-1;  inc(cx);
    while  rstr[cb] = ' '  do  dec(cb);
    len := cb - ca + 1;
    Move(rstr[ca],head[1],len);
    head[0] := Chr(len);
    end;
END;
```

B4

```
PROCEDURE  GetType(nfd : Byte; struct : StructT);
VAR   fd,yy : Byte;  ch : Char;
BEGIN
  WriteLn('Data type:- S = String, C = Char, I : Integer,
                  + ' R = Real, N = Number');
  WriteLn;  yy := WhereY;
  FOR  fd := 1 TO nfd  DO
  WITH  struct^[fd]  DO
    BEGIN
    GotoXY(1,yy);  Write(head);
    GotoXY(40,yy);
    GetOption('Enter data type',['S','C','I','R','N'],dt);
    CASE  dt  OF
     'S' : leng := spc;
     'C' : leng := 1;
     'I' : leng := 4;
     'R' : begin  leng := 6;  Write(' ');
                  GetOption('How many decimal number',[],ch);
                  ReadLn(deci)  end;
     'N' : leng := 0;
    END;  { case }
    inc(yy);
    END;
END;
```

โดยนำชื่อฟิลด์มาเอาต์พุตออกไป แล้วถามว่าเป็นข้อมูลแบบใด อ่านคำตอบมาเก็บไว้ใน dt ของแต่ละฟิลด์โดยตรง สำหรับแบบข้อมูลในโครงงานนี้ได้กำหนดไว้เพียง 5 แบบเท่านั้นคือ

- S หมายถึงสตริง จึงให้ค่าความยาว leng ของข้อมูลเท่ากับค่าความกว้างของช่องคือ spc ซึ่งเหมาะสำหรับภาษาอังกฤษ สำหรับภาษาไทยควรเพิ่มอีกประมาณ 30%

- C หมายถึงอักขระ ให้ leng เป็น 1

- I หมายถึงเลขจำนวนเต็ม ให้ leng เป็น 4 ซึ่งเป็นขนาดข้อมูลของ LongInt เพื่อให้ผู้ใช้สามารถกำหนดตัวเลขได้ถึง บวก-ลบ 2,000 ล้านเศษ

- R หมายถึงเลขจำนวนจริง ให้ leng เป็น 6 ตามขนาดของข้อมูลแบบ Real แล้วถามต่อไปว่าต้องการทศนิยมกี่ตำแหน่ง อ่านมาเก็บใน deci

- N หมายถึง ที่ หากผู้ใช้กำหนดฟิลด์นี้จะนำลำดับที่ของเรคอร์ด (นับจาก 1) ออกแสดง จึงไม่มีการเก็บข้อมูลจริงสำหรับฟิลด์นี้ ให้ leng เป็น 0

หากต้องการอ่านไฟล์ในภาษาปาสคาลจริง จะต้องกำหนดแบบข้อมูลพื้นฐานทั้งหมดครับ

เมื่อได้แบบข้อมูลแล้วขอย้อนกลับไปในโพรซีเดอร์ GetStruct รูป B5 อีกครั้ง ตอนท้ายเป็นการหาค่า offs ของแต่ละฟิลด์ สำหรับค่า recsi คือขนาดของเรคอร์ด เช่นจากตัวอย่างที่ยกมาในตอนต้นคือ 49 ซึ่งค่านี้จะต้องเก็บไว้สำหรับคำนวณหาค่า idx

อ่านจากไฟล์

ขณะนี้ผมยังคงกล่าวถึงว่าเราจะอ่านไฟล์ข้อมูลในภาษาปาสคาลได้อย่างไร แต่เพื่อไม่ให้ซ้ำซ้อน ผมจึงนำโปรแกรมย่อยในโปรแกรม ForYou มาอ้างอิง

สำหรับการอ่านไฟล์ขอให้ดูในโพรซีเดอร์ LoadData รูป C2 ตอนแรกเป็นการตรวจว่าได้เคยโหลดไฟล์ข้อมูลที่อ้างใน finame ลงหน่วยความจำแล้วหรือยัง และมีไฟล์นี้ในดิสค์หรือไม่

ที่โหลดลงหน่วยความจำจริง ๆ อยู่ตอนท้าย แต่ต้องดูตั้งแต่ Reset (fi, 1) ค่า 1 นี้มีความหมายมาก คือหมายความว่าเรคอร์ดของเรามีขนาด 1 ไบต์ ถ้าจะให้ตรงกับการอ่านเรคอร์ดในภาษาปาสคาลจริง ๆ จะต้องใช้คำสั่งเป็น Reset (fi, recsi) โดยที่ recsi คือขนาดของเรคอร์ดเป็นไบต์ ตามตัวอย่างที่ยกมาคือ 49

ที่กำหนดเรคอร์ดเป็น 1 ด้วยเหตุผลเฉพาะของโปรแกรม ForYou ที่จะกล่าวต่อไป เมื่อกำหนดเช่นนี้ค่า size ก็จะเป็นขนาดของไฟล์เป็นไบต์ ที่จะโหลดด้วย BlockRead ให้แก่ data ซึ่งในโปรแกรมหลักได้กำหนดหน่วยความจำให้ถึง 65521 ไบต์ น่าจะพอสำหรับงานเล็ก ๆ

การอ่านจากไฟล์ก็มีเพียงเท่านี้ ขณะนี้ได้ข้อมูลในไฟล์ทั้งหมด (ถ้าไม่เกิน 65521 ไบต์) ในตัวแปร data สำหรับ GetParm เป็นเรื่องของโปรแกรม ForYou

นำไปใช้งาน

เมื่อได้ข้อมูลมาแล้วก็สุดแต่ว่าจะนำไปดำเนินการใด สำหรับในโปรแกรม ForYou จะนำข้อมูลทั้งเรคอร์ดแต่ทีละฟิลด์ไปเก็บในตัวแปรสตริง line ต่อ ๆ กันไป เพื่อที่จะได้เอาต์พุตออกมาทีละ 1 เรคอร์ดเป็น 1 บรรทัด จึงจะขอยกตัวอย่างนี้

ดังที่ดำเนินการด้วยโพรซีเดอร์ OutData รูป C4 ซึ่งเขียนไว้สำหรับจะให้เอาต์พุตค่าในเรคอร์ดใดก็ได้จาก stt ถึง stp มีรายละเอียดดังนี้

- ที่ใช้ค่า rec เป็น rec-1 เพราะผู้ใช้จะอ้างถึงเรคอร์ดโดยเริ่มจาก 1 แต่โปรแกรมเริ่มจาก 0 หาค่าให้แก่ idx ดังกล่าวมา ค่า datstt เป็นเรื่องของโปรแกรม ForYou หากใช้อ่านไฟล์ของปาสคาลแท้ ๆ ให้ตัดออก

- ในวงรอบ FOR ที่อ้างถึง WITH Struct ก็เพื่อนำค่าที่เก็บไว้ในโครงสร้างข้อมูลของแต่ละฟิลด์มาใช้ สำหรับตรงนี้ได้แก่ offs กับ spc

- ได้ให้ค่า idxx ตามที่กล่าวมา

C2

```
PROCEDURE  LoadData;
VAR     ch : Char;  size,res : Word;
        fi : File;
BEGIN
  IF  finame <> ''  THEN
    begin
    Write('Old file is ',finame,'  ');
    GetOption('Work with old file (Y/N)',['Y','N'],ch);
    if  ch = 'Y'  then  Exit  else ClearPointer;
    end;

  WriteLn;
  GetFileName(finame);  Assign(fi,finame);
  {$I-}  Reset(fi,1);  {$I+}
  IF  IOresult <> 0  THEN
    begin  WriteLn('  File not found, press ENTER');
           ReadLn;  finame := '';  stop := True  end
  ELSE
    begin  size := FileSize(fi);
           BlockRead(fi,data^,size,res);
           GetParm  end;
END;
```

ทำให้ idxx ชี้ไปที่ต้นฟิลด์ แล้วย้ายข้อมูลตรงเรคอร์ดนี้ฟิลด์นี้ไปให้แก่ line ด้วย TransferToLine

- เมื่อกลับมาให้กั้นด้วยขีดตั้ง

- ดำเนินการเช่นนี้จนครบทุกฟิลด์ แล้วให้เอาต์พุตด้วย OutLine ขอให้ดูผลการรันในรูปที่ 4 ประกอบ

การทำงานของ TransferToLine ในรูป C3 เป็นการย้ายข้อมูลจาก data ไปให้แก่ line ตามแบบของข้อมูลนั้น ๆ กล่าวคือ สำหรับข้อมูลที่เป็น

- สตริง จะลอกไปตรง ๆ หากอยู่ในโปรแกรมอื่นสามารถเอาต์พุตค่าใน line ได้โดยตรง

- อักขระ จะย้ายข้อมูลไปอยู่ตรงกึ่งกลางช่องที่ผู้ใช้กำหนด

- เลขจำนวนเต็มและเลขจำนวนจริง จะลอกข้อมูลไปให้ item หรือ rtem ซึ่งเป็นตัวแปรแบบเลขจำนวนเต็มและเลขจำนวนจริงตามลำดับ เพื่อบอกให้คอมพายเลอร์ทราบว่าเป็นข้อมูลแบบใด หากอยู่ในโปรแกรมอื่นสามารถเอาต์-พุตค่าใน item หรือ rtom ได้เลย

แต่เนื่องจากผมต้องการนำไปไว้ในสตริง line จึงต้องแปลงเลขจำนวนเป็นสตริงของตัวเลขด้วย Str ได้ค่าใน tstr ก่อน โดยให้ได้สตริงชิดขวาของช่องที่ผู้ใช้กำหนดและจำนวนทศนิยมตามที่ต้องการดังรูปที่ 4 จากนี้จึงลอกไปให้แก่ line อีกทอดหนึ่ง

-ที่ จะแปลงลำดับที่ของเรคอร์ดใน num เป็นสตริงของตัวเลข แล้วลอกไปไว้ใน line เช่นกัน

เท่าที่กล่าวมานี้เป็นการอ่านไฟล์ข้อมูลในภาษาปาสคาล (มีเรื่องของ ForYou ปนอยู่บ้าง) มาดำเนินการ ซึ่งสุดแต่ว่าคุณจะไปดำเนินการอะไร และจะมีประโยชน์หรือไม่เป็นอีกเรื่องหนึ่งที่ผมนำมากล่าวเพราะเป็นส่วนหนึ่งของ ForYou ซึ่งจะต้องกล่าวอยู่แล้ว เพียงแต่นำมากล่าวก่อนสำหรับผู้ที่สนใจในเรื่องนี้

สำหรับค่าในตัวแปร struct เป็นเรื่องที่น่าคิดว่าจะเก็บไว้หรือไม่ ถ้าไม่เก็บก็ต้องกำหนด struct กันทุกครั้งที่อ่าน และ struct หนึ่ง ๆ อาจใช้ได้กับหลายไฟล์ที่มีโครงสร้างข้อมูลเหมือนกัน หากต้องการเก็บก็ให้กำหนดตัวแปรไฟล์เป็น FILE OF FieldT

กลับมาที่โปรแกรม For

จากที่กล่าวมาแต่แรกว่าต้องการให้ใช้ง่าย วิธีการง่าย ๆ คือทำเป็นรายการ

C4

```
PROCEDURE  OutData(stt,stp : Word);
VAR   idx,idxx : Word;  rec,fd,cx,cc,dlen : Byte;
BEGIN
  FOR  rec := stt TO stp  DO
    BEGIN
    idx := datstt + recsi * (rec - 1);
    FillChar(line,80,' ');  cx := 1;
    FOR  fd := 1 TO numfd  DO
    WITH  struct^[fd]  DO
      begin   idxx := idx + offs;
              TransferToLine(rec,idxx,fd,cx);
              cx := cx + spc + 1;
              line[cx-1] := '|'  end;
    line[0] := Chr(cx-1);  OutLine(line);
    END;
END;
```

```
                                                          SOMSUAN Co.
                          CLIENT MEMO
==============================================================================
No |          Name            | age | weight |  tele  |        remark
==============================================================================
  1|Nongrak Sandee            |   25|   45.50|123-456|ok.
  2|Sopha Panngam             |   32|   55.70|222-123|ok.
==============================================================================
```

รูปที่ 4 ↑
แสดงผลการทำงานของโปรแกรมนี้เมื่อให้แสดงผลแต่ละบรรทัด คือที่ให้ OutLine ให้เอาต์พุต

```
PROCEDURE  TransferToLine(num,ind : Word; fd,cx : Byte);
VAR     item : LongInt;  rtem : Real;
        tstr : String[20];  mid : Byte;
BEGIN
  WITH  struct^[fd]  DO
    BEGIN
    CASE  dt  OF
     'S' : Move(data^[ind+1],line[cx],data^[ind]);
     'C' : begin  mid := spc div 2;
                  line[cx+mid] := Chr(data^[ind])  end;
     'I' : begin  Move(data^[ind],item,leng);
                  Str(item:spc,tstr)  end;
     'R' : begin  Move(data^[ind],rtem,leng);
                  Str(rtem:spc:deci,tstr)  end;
     'N' : Str(num:spc,tstr);
    END;   { CASE }
    if  dt in ['I','R','N']  then
      Move(tstr[1],line[cx],Length(tstr));
    END;
END;
```

C3

ให้เลือก ซึ่งการดำเนินการกับไฟล์เห็นว่ามีเรื่องที่จะให้เลือกดังนี้

1. สร้างไฟล์ใหม่
2. แก้ไฟล์เก่า
3. ต่อเติมไฟล์เก่า
4. ค้นหาข้อมูล
5. แสดงผลทางจอภาพ
6. พิมพ์ผล
7. เก็บข้อมูลลงไฟล์

และดังกล่าวแล้วว่าผู้ใช้ไม่ต้องการอ่านเอกสารใดเลยก็จะต้องมีคำอธิบาย แต่ถ้าให้แสดงคำอธิบายทุกครั้งก็คงจะเบื่ออีก จึงกำหนดไว้เป็นรายการให้เลือกได้รายการทั้งหมดตามรูป A3 ขอให้ดูประกอบกับโปรแกรมหลักในรูป M

ในโปรแกรมหลักที่เขียนไว้เป็น begin end คือส่วนที่จะขอให้เขียนเพิ่มเติมกันเองครับ โดยผมจะกล่าวเฉพาะเนื้อหา

การจัดเก็บข้อมูล

จากที่กล่าวมาได้เก็บข้อมูลไว้กับตัวแปร data และโครงสร้างข้อมูลไว้กับตัวแปร Struct เพื่อการใช้งานที่สะดวกจะต้องเก็บโครงสร้างข้อมูลลงไฟล์ด้วย ปัญหาคือจะเก็บไว้ในไฟล์เดียวกันหรือคนละไฟล์ ขอให้ลองคิดดูครับ สำหรับผมเห็นว่าหากเก็บไว้ในไฟล์เดียวกันจะสะดวกต่อผู้ใช้มากกว่า

และเพื่อความสะดวกในการเขียนโปรแกรม ผมเห็นว่าควรจะเก็บเรียงกันไว้ ตั้งแต่อยู่ในหน่วยความจำ โดยให้เก็บค่าในตัวแปรต่าง ๆ ไว้ที่ต้นอะเรย์ data ก่อนที่จะเก็บลงไฟล์ ดังแสดงในรูปที่ 5 ได้แก่

- numti คือจำนวนบรรทัดของไตเติล
- numfd คือจำนวนฟิลด์
- recsi คือขนาดของเรคอร์ด ซึ่งก็คือค่า leng ของทุกฟิลด์รวมกัน
- numrec คือจำนวนเรคอร์ดที่ผู้ใช้ป้อน เช่น ป้อนรายการสิ่งของ 5 รายการ ค่า numrec ก็จะเป็น 5
- strstt หมายถึงตำแหน่งไบต์แรกของข้อมูลใน struct ที่จะนำมาเก็บในอะเรย์นี้
- datstt หมายถึงตำแหน่งไบต์แรกของข้อมูลแท้ ๆ
- TITLE คือพื้นที่ที่จะเก็บข้อความที่เป็นไตเติลจากตัวแปร title
- STRUCT คือพื้นที่ที่จะเก็บข้อมูลของตัวแปร struct
- DATA คือพื้นที่ที่จะเก็บข้อมูลแท้ ๆ ที่ผู้ใช้ป้อน

ค่าต่าง ๆ ดังกล่าวมานี้รวมทั้ง title และ struct ด้วยขอรวมเรียกว่าค่าพารามิเตอร์ในการอ้างถึงต่อไป และเมื่อโหลดจากไฟล์ลงหน่วยความจำให้แก่ตัวแปร data ก็จะนำค่านี้ไปให้แก่ตัวแปรต่าง ๆ ดังกล่าวมาด้วย

จากการจัดเก็บข้อมูลดังกล่าวมานี้ ทำให้ไฟล์ของโปรแกรม ForYou เป็นไฟล์ที่ไม่มีรูปแบบ มีข้อมูลขนาด 1 ไบต์

A3

```
PROCEDURE  DisplayMainMenu;
BEGIN
  Window(1,1,80,25);
  ClrScr;  Window(20,10,60,21);
  WriteLn('MAIN MENU':24);
  WriteLn('=========':24);
  WriteLn('1 Creat a new file');
  WriteLn('2 Correct old data');
  WriteLn('3 Add new data');
  WriteLn('4 Search data');
  WriteLn('5 Display data');
  WriteLn('6 Print data');
  WriteLn('7 Save data');
  WriteLn('I Get Information');
  WriteLn('Q Quit');
  Window(1,1,80,25);
  GotoXY(15,21);
  Write('Your option is ==> ');
END;
```

M

```
var  mch : Char;

BEGIN  { MAIN }
  finame := '';  struct := nil;  title := nil;  sum := nil;
  FillChar(verline,80,'=');  save := True;
  AssignCRT(outfi);  Rewrite(outfi);
  GetMem(data,MAXBUF);


  REPEAT
    DisplayMainMenu;  stop := False;
    GetOption('',['1'..'7','I','Q'],mch);
    WriteLn;
    CASE  mch OF
     '1' : CreateNewFile;
     '2' : begin  end;          { CorrectOldData }
     '3' : AddNewData;
     '4' : begin  end;          { SearchData }
     '5' : DisplayData;
     '6' : PrintData;
     '7' : begin  end;          { SaveData }
     'I' : begin  end;          { Information }
     'Q' : DoQuit;
    END;  { case }
  UNTIL  mch = 'Q';

write('end of program');  readln;
END.
```

numti	numfd	recsi	numrec	strstt	datstt	TITLE	STRUCT	DATA

รูปที่ 6 แสดงการเก็บค่าต่างในตัวแปร data

บ้าง 2 ไบต์บ้าง สตริงบ้าง เรคอร์ดที่ต่างแบบกันบ้าง จึงกำหนดให้เป็นไฟล์ของข้อมูลแบบเรคอร์ดขนาด 1 ไบต์ ดังที่ใช้คำสั่ง Reset (fi, 1)

โปรแกรมสนับสนุน

ก่อนที่จะกล่าวถึงเรื่องราวของโปรแกรม ForYou ต่อไป ขอกล่าวถึงโปรแกรมสนับสนุนที่ผมนำมาใช้ในโครงร่างโปรแกรม มีอยู่ด้วยกัน 3 โปรแกรมอยู่ในรูป A2 คือ

OutLine

นำข้อมูลในสตริงไปออกจอภาพหรือเครื่องพิมพ์ ขึ้นอยู่กับว่าโปรแกรมที่เรียกมาจะให้ outfi เป็นจอภาพหรือเครื่องพิมพ์

GetOption

โปรแกรมนี้จะแสดงข้อความใน astr ถ้าไม่เป็นสตริงว่างที่บรรทัดบนสุดของจอภาพ (หน้าต่าง) และถ้า opt ไม่ใช่เซตว่างจะตรวจจับการกดคีย์ จนกว่าจะได้คีย์ที่อยู่ในเซตของอักขระที่ส่งมาให้แก่ opt แสดงภาพอักขระของคีย์นั้นตรงตำแหน่งเคอร์เซอร์ก่อนที่จะเรียกใช้มัน และให้ค่าคีย์คืนไปใน ch

GetFunctKey

จุดประสงค์หลักคือตรวจจับคีย์หน้าที่พิเศษที่อยู่ในเซต opt2 แต่ให้ตรวจคีย์อื่นได้ด้วยหากส่งมาในเซต opt 1 ให้ค่าคีย์กลับไปใน ch โดยไม่บ่งว่าเป็นคีย์พิเศษหรือไม่การใช้จึงต้องระวังอย่าให้รหัสเหมือนกัน

สร้างไฟล์ใหม่

โปรแกรมย่อยสำหรับรายการต่าง ๆ ดังกล่าวมาต่างก็เป็นอิสระต่อกัน เพียงแต่ใช้โปรแกรมย่อยรอง ๆ ลงไปร่วมกันเท่านั้น จึงขอนำโปรแกรมย่อยในการสร้างไฟล์ใหม่มากล่าวถึงก่อน เพราะตรงข้ามกับการอ่านจากไฟล์ดังกล่าวมาแล้ว

โปรแกรมที่ทำงานในเรื่องนี้คือ CreateNewFile รูป B11 ซึ่งจะมีการดำเนินการ ดังนี้

เตรียมโครงสร้างข้อมูล

ด้วย PrepDataStructure รูป B7 โดยจะอ่านชื่อไฟล์เข้ามาเทียบดูว่าเป็นไฟล์ชื่อเดิมหรือไม่ เพราะการทำงานในโปรแกรม ForYou ผู้ใช้อาจดำเนินการสลับไปมากับไฟล์ข้อมูลต่าง ๆ แล้วตรวจดูว่ามีไฟล์ชื่อนี้ในดิสก์หรือไม่ เพราะ

A2

```
PROCEDURE  OutLine(line : Str79);
BEGIN
  WriteLn(outfi,line);
END;


PROCEDURE  GetOption(astr : Str79; opt : CharSet;
                                   var ch : Char);
VAR   xx,yy : Byte;
BEGIN
  xx := WhereX;  yy := WhereY;
  if  astr <> ''  then
    begin  GotoXY(1,1);  ClrEOL;  Write(astr)  end;
  if  opt <> []  then
    begin
    repeat  ch := UpCase(readKey)  until  ch  in opt;
    GotoXY(1,1);  ClrEol;  GotoXY(xx,yy);  Write(ch)
    end
  else  GotoXY(xx,yy);
END;


PROCEDURE  GetFunctKey(opt1,opt2 : CharSet;var ch : Char);
BEGIN
  repeat
    ch := ReadKey;
    if  ch in opt1  then  exit;
    if  ch = #0  then  ch := ReadKey;
  until  ch in opt2;
END;
```

B11

```
PROCEDURE  CreateNewFile;
VAR   ch : Char;
BEGIN
  ClrScr;  GotoXy(1,3);
  PrepDataStructure;  if  stop  then  Exit;
  Window(1,WhereY,80,25);  numrec := 0;
  GetData(numrec);  Window(1,1,80,25);
  StoreParm;  save := False;
  GetOption('Save or not (Y/N)?',['Y','N'],ch);
  if  ch = 'Y'  then  SaveData;
  Window(1,1,80,25);
END;
```

B7

```
PROCEDURE  PrepDataStructure;
VAR   newname : Str79;  ch : Char;  fi : File;
BEGIN
  GetFileName(newname);
  IF  newname = finame  THEN
    begin
    GetOption('Cancle old data (Y/N)',['Y','N'],ch);
    if  ch = 'N'  then  begin  stop := True;  Exit end;
    end;

  finame := newname;  Assign(fi,finame);
  {$I-}  Reset(fi);  {$I+}
  if  IOresult = 0  then
    begin
    GetOption('Overwrite (Y/N)',['Y','N'],ch);
    if  ch = 'N'  then  begin  stop := True;  Exit end;
    end;

  WriteLn;  GetStruct;
  ClrScr;  GotoXY(1,3);
  OutTitleAndHead;
END;
```

การสร้างไฟล์ใหม่จะยกเลิกข้อมูลในไฟล์เดิม เสร็จแล้วให้ผู้ใช้ป้อนโครงสร้างข้อมูล

- การป้อนโครงสร้างข้อมูล ดำเนินการด้วยโพรซีเดอร์ GetStruct (รูป B5) ตามกรรมวิธีดังกล่าวมา แต่มีบางเรื่องที่ยังไม่ได้กล่าวถึง คือ

- ClearPointer รูป B1 จะยกเลิกหน่วยความจำที่ struct และ title ชี้อยู่เดิม และให้พอยน์เตอร์ทั้งสองนี้มีค่าเป็น nil สำหรับใช้เป็นเงื่อนไขต่อไป

- GetTitle รูป B2 เพื่อให้ผู้ใช้ป้อนข้อความที่ผมเรียกเป็นไตเติล เช่น ชื่อบัญชีหรือตาราง เป็นต้น โดยสอบถามก่อนว่าต้องการไตเติลกี่บรรทัด นำมากำหนดหน่วยความจำให้แก่ title เป็นอะเรย์จำนวนบรรทัดของสตริงขนาด 79 อักขระ

B1

```
 ROCEDURE  GetFileName(var fnme : Str79);
VAR   xx,yy : Byte;
BEGIN
  Write('Enter file name ');  xx := WhereX;  yy := WhereY;
  ReadLn(fnme);  GotoXY(xx+Length(fnme)+2,yy);
END;


PROCEDURE  ClearPointer;
BEGIN
  if  title <> nil  then
    begin  FreeMem(title,80*numti);  title := nil  end;
  if  struct <> nil  then
    begin  FreeMem(struct,SizeOf(FieldT)*numfd);
           struct := nil  end;
END;
```

B2

```
PROCEDURE  GetTitle(var tit : TitleT; var nti : Byte);
VAR     num,xx,yy,cx,len,si : Byte;    ch : Char;
        tem : Str79;
BEGIN
  Write('How many line of title ');  ReadLn(nti);
  if  nti <> 0  then
    begin  GetMem(tit,80*nti);  WriteLn('Enter title') end;

  yy := WhereY;
  FOR  num := 1 TO nti  DO
    BEGIN
    GotoXY(1,yy);  Write(num,' ');
    ReadLn(tem);  len := Length(tem);  GotoXY(60,yy);
    GetOption('Justify Left or Center or Right (L/C/R)',
              ['L','C','R'],ch);
    Write('  ');  FillChar(tit^[num],80,' ');

    if  ch = 'R'  then
      begin  GetOption('From column?',[],ch);
             si := 79;  ReadLn(cx) end
    else if  ch = 'C'  then
      begin  cx := (79-len) div 2;  si := cx + len  end
    else  begin  cx := 1;  si := len  end;

    Move(tem[1],tit^[num,cx],len);
    tit^[num,0] := Chr(si);  inc(yy);
    END;
  WriteLn;
END;
```

B6

```
PROCEDURE  OutTitleAndHead;
VAR  t,fd,cx,cc,len : Byte;
BEGIN
  for  t := 1 to numti  do  OutLine(Title^[t]);
  OutLine(verline);  cx := 1;  FillChar(line,79,' ');
  FOR  fd := 1 TO numfd  DO
  WITH  Struct^[fd]  DO
    begin
    len := Length(head);  cc := cx + (spc-len) div 2;
    Move(head[1],line[cc],len);
    cx := cx + spc + 1;  line[cx-1] := '|';
    end;
  line[0] := Chr(cx-1);
  OutLine(line);  OutLine(verline);
END;
```

เสร็จแล้วให้อ่านข้อความที่ผู้ใช้ป้อนเข้ามาทีละบรรทัด แล้วถามว่าจะให้ข้อความนี้อยู่ชิดซ้าย กึ่งกลางหรือชิดขวา หากชิดขวาจะถามต่อว่าจะให้เริ่มจากคอลัมน์ใด แล้วลอกข้อความที่อ่านมาไปให้แก่ title บรรทัดนั้น ๆ ตามแต่ว่าจะให้อยู่ส่วนไหนของบรรทัด

- การดำเนินการต่อไป คือแยกชื่อฟิลด์ ถามถึงแบบข้อมูลด้วยโพรซีเดอร์ GetHead (รูป B3) และ GetType (รูป B4) แล้วกลับไปหาค่า offs กับ recsi ในโพรซีเดอร์ GetStruct ดังกล่าวมา

การดำเนินการถึงจุดนี้จะได้ค่าต่าง ๆ ที่เรียกว่า พารามิเตอร์เกือบหมด ขาด numrec ซึ่งจะได้ค่านี้ต่อเมื่อผู้ใช้ป้อนข้อมูลแล้ว จึงยังไม่ได้นำไปเก็บไว้กับ data ขอให้สังเกตการหาค่า strstt และ datstt ในรูป B5 ด้วย

- การอ่านชื่อไฟล์ ได้จัดโพรซีเดอร์ GetFileName ไว้ต่างหาก เผื่อคุณจะให้คำอธิบายแก่ผู้ใช้ถึงเรื่องชื่อไฟล์ตามข้อกำหนดของระบบจัดการ และเพื่อไม่ให้เคอร์เซอร์ขึ้นบรรทัดใหม่ ดังแสดงในรูป B1

แสดงตาราง

ก่อนที่จะให้ผู้ใช้ป้อนข้อมูล จะเอาต์พุตข้อความที่เป็นไตเติลและส่วนหัวของตารางด้วย OutTitleAndHead รูป B6 โดยเอาต์พุตข้อมูลใน title ออกไปทีละบรรทัดตรง ๆ

สำหรับชื่อฟิลด์จะลอกลง line ทีละฟิลด์โดยลอกให้มาอยู่ตรงกึ่งกลางของช่องที่ผู้ใช้กำหนดไว้ ซึ่งคั่นด้วยเครื่องหมายขีดตั้ง แล้วเอาต์พุตค่าใน line ออกไปทั้งบรรทัด

อ่านข้อมูล

ขั้นตอนต่อไปคือให้ผู้ใช้ป้อนข้อมูล ได้เปิดหน้าต่างใต้ส่วนหัวของตาราง เพื่อที่จะให้ข้อมูลที่ผู้ใช้ป้อนเลื่อนอยู่ใต้ส่วน

หัวของตาราง แล้วเรียกใช้ GetData เพื่ออ่านข้อมูลที่ผู้ใช้ป้อนมาเก็บใน data โดยเริ่มจากเรคอร์ดที่ 0 จนกว่าผู้ใช้จะกดคีย์ Enter แทนการป้อนข้อมูล

การทำงานของ GetData รูป B8 จะหาค่าให้ idx และ idxx ดังกล่าวมา ในวงรอบ FOR ซึ่งเป็นการดำเนินการทีละฟิลด์นั้น จะตรวจว่าถ้าเป็นข้อมูลแบบ "ที่" จะเอาต์พุตค่าเรคอร์ด-ที่ออกไป ถ้าเป็นข้อมูลแบบอื่นจะอ่านข้อมูลเข้ามาเป็นสตริงเก็บใน tstr แล้วตรวจว่าหากเป็นสตริงว่างจะออกจากโปรแกรมย่อยนี้ เป็นการยุติการป้อนข้อมูล

หากไม่ว่างจะตรวจต่อไปว่าเป็นข้อมูลแบบใด ซึ่งถ้าเป็นเลขจำนวนจะเปลี่ยนเป็นสตริงของตัวเลขก่อน แล้วลอกไปลง data ณ ตำแหน่ง idxx ด้วยวิธีนี้ทำให้เราสามารถอ่านข้อมูลทีละฟิลด์ทีละเรคอร์ดที่ผู้ใช้ป้อนไปเก็บในหน่วยความจำได้

เก็บพารามิเตอร์

จากนี้จะเป็นการเก็บค่าพารามิเตอร์ในตัวแปรต่าง ๆ ไว้กับ data ด้วย Store Parm รูป B9 ซึ่งเป็นการลอกข้อมูลไปไว้ใน data ตามตำแหน่งที่กำหนดและตามขนาดของตัวแปรนั้น ๆ

เก็บลงไฟล์

หากผู้ใช้ต้องการให้เก็บลงไฟล์จะดำเนินการด้วย SaveData รูป B10 ซึ่งดำเนินการกลับกับ LoadData ที่เคยกล่าวแล้ว

เป็นการจบการทำงานของการสร้างไฟล์ใหม่

B8

```
PROCEDURE  GetData(var recnum : Word);
VAR    xx,yy,topY,fd : Byte;   idx,idxx : Word;
       tstr : Str79;  item : LongInt;  rtem : Real;
       res : Integer;
BEGIN
  topY := WhereY;
  REPEAT
    idx := datstt + recsi * recnum;
    xx := 1;  yy := WhereY;
    if  yy = (26 - topY)  then
      begin  WriteLn;  yy := yy - 1   end;
    FOR  fd := 1 TO numfd  DO
    WITH  struct^[fd]  DO
      BEGIN
      idxx := idx + offs;  GotoXY(xx,yy);
      if  dt = 'N'  then  Write(recnum+1:spc)
      else  begin  ReadLn(tstr);
                   if  tstr = ''  then  Exit  end;

      if  dt = 'S'  then
        Move(tstr[0],data^[idxx],Length(tstr)+1)
      else if  dt = 'C'  then  data^[idxx] := Ord(tstr[1])
      else if  dt = 'I'  then
        begin  Val(tstr,item,res);
               Move(item,data^[idxx],leng)  end
      else if  dt = 'R'  then
        begin  Val(tstr,rtem,res);
               Move(rtem,data^[idxx],leng)  end;

      xx := xx + spc + 1;  GotoXy(xx-1,yy);  Write('|');
      END;
    inc(recnum);  WriteLn;
  UNTIL  tstr = '';
END;
```

B9

```
PROCEDURE  StoreParm;
BEGIN
  data^[NTI] := numti;  data^[NFD] := numfd;
  Move(recsi,data^[RSI],2);
  Move(numrec,data^[NREC],2);
  Move(strstt,data^[SS],2);
  Move(datstt,data^[DS],2);
  Move(title^,data^[TIT],80*numti);
  Move(struct^,data^[strstt],SizeOf(FieldT)*numfd);
END;
```

B10

```
PROCEDURE  SaveData;
VAR    size,res : Word;  fi : File;
BEGIN
  Assign(fi,finame);  ReWrite(fi,1);
  size := datstt + numrec * recsi;
  BlockWrite(fi,data^,size,res);
  Close(fi);  save := True;
END;
```

C5

```
PROCEDURE  DisplayData;
VAR    stt,stp : Word;  topY,difY : Byte;  ch : Char;

  procedure  AddStp(frm : Byte);
  begin
    stp := frm + dify;
    if  stp > numrec  then  stp := numrec;
  end;

  procedure  MinusStt(frm : Byte);
  var  tem : Integer;
  begin
    tem := frm - dify;
    if  tem < 1  then  stt := 1  else stt := tem;
  end;

BEGIN
  LoadData;  if  stop  then  Exit;
  ClrScr;  OutTitleAndHead;
  topY := WhereY;  dify := 25 - topY - 1;
  Window(1,topY,80,25);  stt := 1;
  AddStp(stt);
  REPEAT
    ClrScr;
    OutData(stt,stp);  GotoXY(1,1);
    GetFunctKey([Esc],[PgUp,PgDn],ch);
    if  ch = PgDn  then
      begin  AddStp(stp);  MinusStt(stp)  end
    else if  ch = PgUp  then
      begin  MinusStt(stt);  AddStp(stt)  end;
  UNTIL  ch = Esc;
  Window(1,1,80,25);
END;
```

แสดงผลทางจอภาพ

เรื่องราวของการแสดงผลทางจอภาพ ได้กล่าวไว้เป็นส่วนใหญ่แล้วในเรื่องการอ่านไฟล์ข้อมูลในภาษาปาสคาล สำหรับในโปรแกรม ForYou ใช้โปรแกรม DisplayData รูป C5 ดำเนินการในเรื่องนี้

เริ่มด้วยการโหลดข้อมูลจากไฟล์ลงดิสก์ด้วย LoadData (รูป C2) ซึ่งจะเรียกใช้ GetParm ในรูป C1 อีกทอดหนึ่ง แสดงไตเติล และส่วนหัวของตารางด้วย OutTitleAndHead (รูป B6) จากนั้นเป็นการนำข้อมูลออกแสดงทีละ 1 จอภาพ โดยให้สามารถเลื่อนหน้าต่อไป (หากมี) และหน้าก่อนนี้ (หากมี) มาดูได้ด้วย PgDn และ PgUp ตามลำดับ ดังเช่นในเอดิเตอร์ของเทอร์โบปาสคาล

C1

```
PROCEDURE  GetParm;
VAR   size : Word;  fd : Byte;
BEGIN
  numti := data^[NTI];  numfd := data^[NFD];
  Move(data^[RSI],recsi,2);
  Move(data^[NREC],numrec,2);
  Move(data^[SS],strstt,2);
  Move(data^[DS],datstt,2);
  size := 80 * numti;  GetMem(title,size);
  Move(data^[TIT],title^,size);
  size := SizeOf(FieldT) * numfd;
  GetMem(struct,size);
  Move(data^[strstt],struct^,size);  size := 0;
  for  fd := 1 to numfd  do
  with  struct^[fd]  do  size := size + spc + 1;
  verline[0] := Chr(size);
END;
```

D

```
PROCEDURE  PrepPrinter;
VAR   reg : Registers;  ch : Char;
BEGIN
  REPEAT
    GotoXY(38,1);
    reg.ah := 1;  reg.dx := 0;  Intr($17, reg);
    IF  (reg.ah <> 16 ) AND (reg.ah <> 8)  THEN
      GetOption('  PRINTER NOT READY  try again (Y/N) ',
                          ['Y','N'],ch)
    ELSE  Exit;
  UNTIL  ch = 'N';
  stop := True;
END;


PROCEDURE  PrintData;
BEGIN
  PrepPrinter;  if  stop  then  Exit;
  LoadData;  if  stop  then  Exit;
  WriteLn;  Write('Output to printer.....');
  Assign(outfi,'PRN');  Rewrite(outfi);
  OutTitleAndHead;  OutData(1,numrec);
  OutLine(verline);
  AssignCRT(outfi);  Rewrite(outfi);
  Write('finish, press enter ');  readln;
END;
```

E

```
PROCEDURE  DoQuit;
VAR   ch : Char;
BEGIN
  if  not save  then
    begin  GetOption(finame+' not saved.  Save(Y/N) ',
                                ['Y','N'],ch);
           if  ch = 'Y'  then  SaveData;
    end;
  WriteLn('   Goodbye');
END;
```

พิมพ์ผล

การพิมพ์ผลทางเครื่องพิมพ์ใช้โปรแกรมย่อยเดียวกันกับการแสดงผลทางจอภาพ โดยเปลี่ยนเอาต์พุตจากจอภาพไปเป็นเครื่องพิมพ์ด้วย Assign (outfi, 'PRN') ดังที่เห็นในโพรซีเดอร์ PrintData รูป D แต่ก่อนที่จะดำเนินการได้ตรวจก่อนว่าเครื่องพิมพ์ว่างหรือไม่ด้วย PrepPrinter และภายหลังที่ได้เอาต์พุตข้อมูลออกทางเครื่องพิมพ์แล้ว ได้เปลี่ยนเอาต์พุตกลับมาเป็นจอภาพตามเดิม

การจบโปรแกรม

เมื่อผู้ใช้เลือกรายการ Q เพื่อยุติการทำงาน หากข้อมูลยังไม่ได้เก็บลงไฟล์ ได้ให้ถามว่าจะให้เก็บลงไฟล์หรือไม่ด้วย DoQuit รูป E

โปรแกรมที่เหลือ

โปรแกรมที่ผมเขียนให้แก่ผู้ต้องการมีเพียงเท่านี้ เพราะจุดใหญ่เพื่อแสดงการอ่านและเขียนกับไฟล์ข้อมูลเท่านั้นครับ ที่จะต้องเขียนเพิ่มคือ

ต่อเติมไฟล์เก่า

โปรแกรมนี้เขียนในแนวเดียวกับการสร้างไฟล์ใหม่ แต่ให้เริ่มด้วย Load Data และเรียกใช้ GetData ด้วยการผ่านค่า numrec ไปให้

ค้นหาข้อมูล

ขึ้นอยู่กับการตัดสินใจครับว่า หากเห็นว่าโปรแกรมที่ใช้ในการแสดงผลทาง

จอภาพ สามารถจะใช้ในการค้นหาข้อมูลได้ด้วย โดยให้ผู้ใช้เลื่อนจอภาพไปมาหาข้อมูลเอง ก็ไม่ต้องดำเนินการใด

แต่ถ้าต้องการให้ผู้ใช้ค้นหาข้อมูลด้วยการป้อนข้อมูลที่ต้องการค้น เช่น ป้อนชื่อ เพื่อให้เครื่องแสดงข้อมูลทั้งเรคอร์ด ก็ควรจะแสดงชื่อฟิลด์ในลักษณะ pull-down menu ให้ผู้ใช้เลือกว่าจะอ้างอิงถึงฟิลด์ใด แล้วให้ป้อนข้อมูลที่ต้องการค้น เช่น ชื่อ

นำค่าฟิลด์มาหาค่า offs ให้แก่ idxx เทียบข้อมูลในเรคอร์ดแรก แล้วเพิ่มค่า idxx ด้วยค่า recsi เพื่อเทียบข้อมูลในเรคอร์ดต่อไปและต่อ ๆ ไป จนพบหรือครบทุกเรคอร์ด หากพบให้แสดงด้วย OutData

แก้ไฟล์เก่า

หากให้ผู้ใช้ค้นหาข้อมูลด้วย DisplayData ก็ควรให้สามารถแก้ข้อมูลได้ด้วยโปรแกรมนี้ด้วย โดยเพิ่มการตรวจจับคีย์ลูกศรทั้งสี่ ซึ่งถ้าผู้ใช้กดคีย์ลูกศรขึ้นหรือลงจะเป็นการลบหรือบวกค่าเรคอร์ด ถ้ากดคีย์ลูกศรขวาหรือซ้ายจะเป็นการลบค่าฟิลด์ ถ้าผู้ใช้กดคีย์ Enter ให้ถือว่าต้องการแก้ฟิลด์นั้นเรคอร์ดนั้น

ให้ผู้ใช้ป้อนข้อมูลใหม่ซึ่งควรเป็นที่บรรทัดแรกของจอภาพอ่านมาเก็บไว้ในสตริง แล้วเปลี่ยนเป็นข้อมูลให้แก่ data ตามแบบของข้อมูลดังกล่าวใน GetData (ควรทำโปรแกรมท่อนนี้เป็นโปรแกรมย่อย) แล้วนำข้อมูลทั้งเรคอร์ดออกแสดงซ้ำในบรรทัดที่แก้ด้วย OutData (จะได้ไม่ต้องเขียนโปรแกรมใหม่)

แต่ถ้าให้ผู้ใช้ค้นหาข้อมูลด้วยการป้อนข้อมูล เมื่อค้นพบแล้วต้องถามต่อว่าจะให้แก้ในฟิลด์ใด เมื่อได้ค่าฟิลด์มาแล้วให้ดำเนินการต่อไปดังกล่าวข้างบน

แนวความคิดเรื่อง Help

ตามที่กล่าวแล้วว่าจะให้ผู้ใช้กดคีย์ I (ซึ่งคุณจะเปลี่ยนเป็น H ก็ได้) เพื่อขอคำอธิบาย ในการนี้คุณจะต้องเตรียมคำอธิบายไว้ในเท็กซ์ไฟล์เพื่อแสดงเมื่อผู้ใช้กดคีย์ I ดังกล่าว หากมีหลายหน้าต้องให้ผู้ใช้เลือกข้อความในหน้าก่อน-หลังมาดูได้ (แนวการเขียนโปรแกรมเช่นนี้ ดูได้ในโพรซีเดอร์ Display Data รูป C5)

แต่ในการใช้งานจริงผู้ใช้อาจเกิดสงสัยขณะที่ทำงานอยู่ในโปรแกรมย่อยใดก็ได้ หากต้องให้กลับมาเมนูหลักนอกจากจะเสียเวลาแล้วจะมีปัญหากับงานที่ผู้ใช้ดำเนินการค้างอยู่ ควรให้ผู้ใช้เรียกดูได้ด้วยคีย์หน้าที่พิเศษเช่น F1

การที่จะให้ดำเนินการเช่นนี้ได้ทำได้หลายวิธี วิธีหนึ่งซึ่งเป็นอัตโนมัติและนำไปใช้ในโปรแกรมอื่นได้ คือ เราต้องอ่านคีย์ด้วยตนเองไม่ใช้คำสั่ง Read เช่น สร้างโพรซีเดอร์ ReadStr และ GetChar โดยให้ ReadStr อ่านสตริงโดยอ่านอักขระมาทีละตัวด้วย GetChar และให้ GetChar ตรวจ ถ้าเป็นคีย์ F1 ให้เรียกใช้โปรแกรมย่อยที่จะแสดงคำอธิบาย (ขอให้ดูแนวการเขียนในเรื่อง เพื่อนพิมพ์ในไมโครฯ ฉบับยูทิลิตี้)

ก็มีเรื่องปลีกย่อยอีกครับ คือในการแสดงคำอธิบายผู้ใช้คงต้องการคำอธิบายเฉพาะตรงเรื่องที่ตนกำลังดำเนินการอยู่ ก็ต้องแยกคำอธิบายเป็นเรื่อง ๆ เมื่อป้อนคงป้อนเรียงติดกันแต่อาจใช้เครื่องหมายบางตัว เช่น * นำหน้าคำอธิบายแต่ละเรื่องไว้ เตรียมตัวแปรเป็นอะเรย์ของ Word เมื่อโหลดคำอธิบายลงหน่วยความจำแล้ว ให้ค้นหา * นำไบต์-ที่ไปให้แก่อะเรย์นี้ เพื่อให้ชี้ตรงคำอธิบายที่ต้องการ (ขอให้ดูในเรื่องภาษา LOGØ ไมโครฯ ฉบับที่ 50 คือที่ให้ค่าแก่ตัวแปร proc)

สำหรับในโปรแกรมให้เตรียมตัวแปรแบบไบต์ไว้ เช่น ให้ชื่อว่า helpnum และในโปรแกรมที่กล่าวมาตรงใดที่คุณเตรียมคำอธิบายไว้ ให้แทรกถ้อยแถลงให้ค่า helpnum เป็นลำดับที่ตามที่กำหนดไว้ในคำอธิบาย และให้โปรแกรมย่อยที่จะแสดงคำอธิบาย นำค่าใน helpnum ไปใช้เป็นจุดเริ่มต้นในการแสดงคำอธิบาย (ที่ผมกล่าวมานี้เป็นเพียงวิธีหนึ่งเท่านั้น อาจมีวิธีดีกว่านี้ครับ)

อนึ่ง ในการแสดงคำอธิบาย จะต้องมีการเปิดหน้าต่าง เก็บภาพบนจอภาพ ฯลฯ ขอให้ดูในหนังสือเรียนรู้ภาษาปาสคาล : ด้วยเทอร์โบปาสคาล 4.0-5.0 บทที่ 11

การปรับปรุง

อีกเรื่องหนึ่งที่ผมยังไม่กล่าวถึงคือการบวกทบเพื่อเป็นยอดรวม แต่ได้เตรียมตัวแปรไว้ให้แล้วคือ sum เป็นอะเรย์ของเลขจำนวนจริง กับ total เป็นอะเรย์ของบูลี ซึ่งคุณจะต้องต่อเติมโปรแกรม กำหนดจำนวนช่องให้เท่ากับจำนวนฟิลด์

และถามผู้ใช้ว่าต้องการยอดรวมในช่องใดบ้าง ให้ค่าความเป็นจริงในอะเรย์ total ช่องนั้น (นอกนั้นให้เป็นเท็จ) แล้วบวกทบค่าในช่องนั้นเก็บไว้ในตัวแปร sum โดยไม่คำนึงว่าเป็นเลขจำนวนจริงหรือจำนวนเต็มในการแสดงผลจึงจะแสดงผลตามแบบข้อมูล

นอกจากนี้ในตารางหากผู้ใช้กำหนดช่อง จำนวน ราคาและเงิน จะให้ผู้ใช้ป้อนข้อมูลในช่องเงินหรือจะให้เครื่องคำนวณให้ หากต้องการให้เครื่องคำนวณให้วิธีหนึ่งคือ กำหนดแบบข้อมูลใหม่เป็น M แล้วถามต่อว่าจะให้ดำเนินการแบบใด เช่น ให้ป้อนเป็น 3 * 4 ต่อ

การใช้งาน

คงเป็นไปตามที่กล่าวมาแต่ต้นคือ

- ทำบัญชีเล็ก ๆ น้อย
- เก็บคำสำหรับ พจนานุกรม ดัชนี นามบัตร ฯลฯ

พัฒนาต่อ

โปรแกรมนี้ผมขอมอบให้แก่ทุกท่าน (หากเห็นสมควรรับ) เมื่อผู้ใดนำไปพัฒนาต่อก็เป็นสิทธิของผู้นั้นครับ (รวมทั้งตัวผมเองด้วย)